Alternative Assurance Criteria

Dr. David Brewer
Gamma Secure Systems Limited
www.gammassl.co.uk
Agenda

- Motivation
- Meta Criteria
- Common Criteria Interpretation
- Alternative Assurance Criteria Interpretation
- Conclusions
Motivation
Reported at recent ICCCs

- “Marketing is finding out what the customer wants”, Marketing panel (Tokyo)

- “After we have developed a product we employ a consultant to create the Common Criteria design documentation”, Microsoft (Tokyo and Berlin)

- There has to be a better way

... and then I remembered
a long, long time ago...

- Before the first CESG evaluation facility (1986)
- Successful evaluations of UK government and banking IT systems
- High assurance gained through formal code analysis techniques (SPADE/MALPAS)
- Some work published but long since forgotten
- Approach abandoned in favour of quasi-harmonisation with Orange Book
Meta Criteria (pre 1986)

A piece of ancient history
Goals

“Correct operation confirmed or weaknesses and countermeasures identified”

In those days we spoke of software integrity (not security):

> Software does what it is supposed to do and not what it is not supposed to do”

Use precluded functions/properties (e.g. covert channels) as well as required functions/properties
Assurance

- Based on the extent of evaluator’s knowledge about the TOE

- Always used a model of the implementation as a means to reason about software integrity

- Always included testing (machine code verification regarded as impractical)

- Always checked quality controls (to ensure evaluating the right thing)

- Principle of escalation

  - Note: Orange Book also has some examples of this
Meta Criteria

- Level of abstraction of the model
  - How far removed from the implementation

- Form of the abstraction
  - What does it allow the analyst to reason about
  - How does it allow the analyst to reason

- Means of model creation
  - Design intent or reverse engineering of the implementation
Creation of the model

- Two principal routes:
  - Implementation route (i.e. use the design documentation)
  - Analysis route (i.e. reverse engineer the models from the final implementation)

- Can be a mix

**KEY:**
- AC: Application concept (e.g. user requirement)
- FS: Formal specification (e.g. top level design)
- PFS: Partial formal specifications (i.e. intermediate representations)
- OS: Operational system (the actual resultant implementation)
- E: Environment
Common Criteria Interpretation
Basic thesis

- Use effectiveness criteria to reason about security
- Use design documentation for the models
- Use correctness criteria to argue that the models are a sufficiently correct representation of the implementation
EALs (V-model view)

- Customer requirements
- High level design
- Low level design
- Code

1. Acceptance testing
2. Integration testing
3. Module testing
4. Program testing
Example 1: GlobalPlatform

- Could have produced PP based on Visa Open Platform PP (OP3) – see ICCC2 (Brighton)
- But too many PPs (OP3, JavaCard, SCSUG, SSVG)
- Wanted something more intelligible
- Way ahead shown by ITRI – see ICCC3 (Ottawa)
- Result was the Card Security Requirements Specification – see www.globalplatform.org
Example 1: GlobalPlatform

- Semi-formal specification covering card content management down to and including the IC (physical)
- Addressed chip card “composition problem”
- Facilitated EAL6 evaluation (at least)
- But card vendors just go for EAL4+
Example 2: Microsoft’s SDL

|-------------------------------------|---------------------------------|--------------------------------------|-----------------|---------------------------------------------------------------|---------------------------------------------|---------------------------------|-------------------------|------------------------|-----------------------------------------|

- Focused on adding steps that reduce vulnerability rates during development
  - Engineer training
  - Threat modeling
  - Coding standards, code reviews
  - Use of static analysis tools
  - Fuzz testing
  - Independent “Final Security Reviews”

- Doesn’t map at all well to correctness requirements
- Evaluation documentation has to be specially produced after the fact
- Unnecessary expense

Steven B. Lipner, Microsoft, ICCC6 (Tokyo)
Alternative Assurance Interpretation
The alternative

- CC assurance uses models derived from design intent

- Alternative: use models reverse engineered from the actual implementation (or a mix)

- Note:
  - less expensive (no need for correctness criteria)
  - more reliable (based on the actual implementation)
  - Higher assurance (analysis will use formal code (logic) analysis methods)
Example 1: GlobalPlatform

- Proposition 1:
  - Animate the Card Security Requirements Specification and verify effectiveness
  - Use the results to generate tests
  - Use these tests to test the TOE

- Proposition 2:
  - As above, plus
  - Analyse reversed-engineered logic modules and confirm results with TOE-specific testing

Note the GlobalPlatform Card Security Requirements Specification, animation and associated tests would be a reusable evaluation resource.
Example 1: GlobalPlatform

Proposition 1:
- Animation and verification
- Derive test case
- Use these to show TOE meets security specification

Proposition 2:
- As 1 plus
- Analyse reverse-engineered logic modules
- Conform results with TOE-specific testing
Example 2: Microsoft’s SDL

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Kickoff &amp; Register with SWI</td>
<td>Security Arch &amp; Attack Surface Review</td>
<td>Threat Modeling</td>
<td>Use of static analysis tools</td>
<td>Fuzz testing</td>
<td>Engineer training</td>
</tr>
</tbody>
</table>

Focused on adding steps that reduce vulnerability rates during development

1. (Semi) formal specification/architecture?
2. Could be used to create formal model of source code
3. Use spec/arch and code models to generate tests

Steven B. Lipner, Microsoft, ICCC6 (Tokyo)
Example 2: Microsoft’s SDL

1. Security Architecture, Attack Surface Review & Threat Modelling?
2. Formal analysis of reverse engineered source code modules
3. Fuzz testing, augmented with tests derived from (1) and (2)
Conclusions
Conclusions

- Our goal is to remove vulnerabilities
- Could create an alternative criteria based on meta-criteria
- Could be defined to yield an equivalence in terms of EAL
- Could never be created by incremental CC development
- More suited to actual development methodologies?
- Higher assurance for less cost?
- Would need to be driven by vendors
- Go for it?
Alternative Assurance Criteria

Any Questions?

Dr. David Brewer
Gamma Secure Systems Limited
www.gammassl.co.uk

©Gamma Secure Systems Limited, 2006